UERJ - UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO
IME - INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
CComp - Programa de Pós-Graduação em Ciências Computacionais
IME02701 Álgebra Linear: Aspectos Teóricos e Computacionais
Prof.ª Cristiane Oliveira de Faria

Relatório nº 1 - Estudo comparativo entre métodos de resolução de sistemas lineares

Aluno: Nathalia de Almeida Castelo Branco
e-mail: nathaliaacbranco@gmail.com

27 de abril de 2021

1 Introdução

Sistemas lineares e matrizes estão presentes nos mais diversos campos de atuação e são especialmente úteis para resolver problemas dos mais diferentes tipos de complexidade, desde um questionamento simples como a quantidade de produto a ser estocada em um depósito para alcançar alguma combinação específica a discretização de estruturas em elementos finitos. A aplicação dos sistemas lineares é ampla e extremamente útil, especialmente com o uso da tecnologia para auxiliar na resolução de problemas.

Os sistemas lineares passaram pela contribuição de diversos matemáticos para chegarmos até o que temos hoje, as notações, os conceitos e os teoremas foram modificados e aperfeiçoados no decorrer do tempo [1].

O objetivo deste trabalho é realizar um estudo comparativo entre métodos de resolução de sistemas lineares para matrizes específicas que serão descritas posteriormente. Os métodos que serão estudados são métodos de resolução diretos e iterativos, sendo eles: Eliminação Gaussiana, Decomposição LU, Decomposição Gauss-Jordan e métodos iterativos de Jacobi e Gauss-Seidel.

Na seção 1, é realizada uma breve introdução do tópico assim como o objetivo do trabalho. Na seção 2, são descritos os conceitos básicos e suficientes para o entendimento da resolução dos sistemas. Na seção 3, são descritos os métodos diretos de resolução de sistemas lineares e os métodos iterativos, e também são expostos os pseudocódigos utilizados no desenvolvimento computacional dos métodos de resolução. Já na seção 4, o problema utilizado para realização do estudo comparativo é descrito. Na seção 5, o equipamento utilizado para realização dos testes computacionais é descrito. Nesta seção também são expostos os resultados obtidos com o desenvolvimento dos algoritmos da seção 3 na linguagem Python, estes resultados são sintetizados em tabelas e gráficos para visualização e conclusão do estudo comparativo.

2 Conceitos básicos

Nesta seção serão definidos os conceitos básicos para o entedimento do problema proposto para este relatório, como sistemas lineares, matrizes e formas de representação.

Um conceito importante é entender a representação dos sistemas lineares. Um sistema de equações lineares de m equações e n incógnitas tem a seguinte forma:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_m + a_{m2}x_2 + ... + a_{mn}x_n = b_m \end{cases}$$

para a_{ij} com $1 \leq i \leq m$ e $1 \leq j \leq n$ pertecente aos números reais ou complexos. Uma solução para este sistema é uma n-upla de números $(x_1, x_2, ..., x_n)$ [2].

Este mesmo sistema pode ser escrito na forma matricial:

$$\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & ... & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Na forma simbólica:

$$A \cdot x = b \tag{1}$$

Onde a matriz A é a matriz dos coeficientes, o vetor x é o vetor das incógnitas e o vetor b é o vetor dos termos independentes.

Uma outra representação para um sistema linear é a chamada **matriz ampliada ou aumentada** do sistema, onde os termos da extrema direita da matriz representam o vetor dos termos independentes, constituindo assim uma representação abreviada das equações do sistema linear [2], conforme podemos ver a seguir.

$$\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} & b_1 \\ a_{21} & a_{22} & ... & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & ... & a_{mn} & b_m \end{bmatrix}$$

Exposta as maneiras de representação de um sistema linear, temos então o objetivo inicial que não consiste em apenas representar o problema e sim, resolvê-lo. Para isto é importante mencionar que visando a eficiência dos cálculos, o sistema em geral é transformado em uma matriz e então são realizadas as chamadas **operações elementares** nesta matriz para que o mesmo seja resolvido.

As operações elementares [2] são feitas sobre as linhas de uma matriz e são:

- Permutação entre linhas ($L_i \leftrightarrow L_j$)
- Multiplicação da i-ésima linha por um escalar k não nulo ($L_i \leftarrow k \cdot L_i$)
- Substituição de uma linha por ela mesma mais n vezes outra linha ($L_i \leftarrow L_i + k \cdot L_j$)

A matriz obtida após a realização de operações elementares é chamada de matriz equivalente B e pode ser representada pela seguinte notação: $A \sim B$. Então, a partir deste conceito é possível resolver sistemas apenas com finitas operações elementares para que sejam eliminadas incógnitas e seja obtida uma nova matriz ampliada equivalente, podendo estar na forma escalonada (triangular) ou escalonada reduzida [2].

Dada uma matriz m x n, a mesma está na **forma escalonada** [3] se :

1. Toda variável líder está na coluna a esquerda da variável líder da linha abaixo
2. Linhas compostas por zeros estão na parte inferior da matriz

Dada uma matriz m x n, a mesma está na **forma escalonada reduzida** [2] se:

1. O primeiro elemento não nulo de uma linha não nula é 1
2. Cada coluna que contém o primeiro elemento não nulo de alguma linha tem todos os seus outros elementos iguais a zero
3. Toda linha nula ocorre abaixo de todas as linhas não nulas
4. O elemento não nulo da linha i ocorre na coluna k_i

Em uma matriz na forma escalonada, são chamados de **pivô** os termos que possuem variável líder, sendo eles números diferentes de zero utilizados para zerar os termos de interesse na matriz [3].

Sobre as **soluções de um sistema linear de equações**, o mesmo é dito consistente quando há solução, podendo esta ser única ou ter infinitas soluções, e inconsistente quando não há solução [4]. No caso do sistema estar na forma escalonada, existem duas situações possíveis [3]:

1. Se o sistema não possui variáveis livres, ele é dito triangular e possui uma única solução

2. Se o sistema possui pelo menos uma variável livre, neste caso, a solução geral possui parâmetros livres e então o sistema tem infinitas soluções

3 Métodos de resolução de sistemas lineares

Entendido os conceitos básicos, nesta seção serão descritos os métodos de resolução de sistemas lineares propostos no enunciado do relatório. São eles: eliminação gaussiana, decomposição LU, decomposição de Gauss-Jordan, método iterativo de Jacobi e método iterativo de Gauss-Seidel. Nesta seção serão também são expostos os algoritmos utilizados na implementação dos métodos de resolução de sistemas lineares.

3.1 Métodos Diretos

3.1.1 Eliminação Gaussiana

Este método foi descoberto por matemáticos chineses há mais de 2000 anos. Seu nome foi dado em homenagem ao matemático alemão Carl Friedrich Gauss, que descobriu o método de forma independente e o tornou conhecido no ocidente no século XIX [3].

O método de eliminação de Gauss consiste essencialmente em duas partes [4]:

1. Eliminação, onde o sistema é reduzido a um sistema triangular ou a uma matriz escalonada através de operações elementares

2. Substituição, a partir da última incógnita do sistema $x_m = b_m$ encontrando as anteriores de forma decrescente

Neste método, dada uma matriz A após o a fase de eliminação, a mesma é reduzida a forma escalonada, então o sistema equivalente é obtido desta matriz e a retrossubstituição das incógnitas é feita para que o vetor solução seja obtido [4]. Para sua utilização não é necessário que a matriz seja quadrada [3].

Para o desenvolvimento computacional deste método, utilizou-se o algoritmo 1 para a primeira parte referente a eliminação.

Algorithm 1: Eliminação Gaussiana

Data: Matriz A_{mxn} e vetor b_m
Result: Matriz A_{mxn} e vetor b_m escalonados

```
for j = 1, ..., n − 1 do
    for i = j + 1, ..., m do
        m_ij = a_ij / m_jj  b_i = b_i − m_ij · b_j  for k = j, ..., n do
            a_ik = a_ik − m_ij · a_jk
        end
    end
end
```

Já para a segunda parte da eliminação gaussiana, chamada de substituição, onde são encontradas as variáveis, o algoritmo 2 foi utilizado.

Algorithm 2: Retrossubstituição

Data: Matriz A_{mxn} e vetor b_m escalonados
Result: Vetor solução x_n

for $i = n, n-1..., 1$ **do**
  $x_i = b_i$
  for $j = i+1, i+2, ..., n$ **do**
    $x_i = x_i - a_{ij} \cdot x_j$
  end
  $x_i = \dfrac{x_i}{a_{jj}}$
end

3.1.2 Decomposição LU

Dada uma matriz A não singular, a mesma pode ser decomposta em duas matrizes triangulares a partir da eliminação $A = LU$. Destas matrizes, a matriz U é a obtida através da eliminação gaussiana. Visando obter a matriz A através da matriz U, precisamos da matriz L, cujos termos consistem nos multiplicadores utilizados para obtenção da matriz U [5].

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ 0 & a_{22} & a_{23} & \cdots & a_{2m} \\ 0 & 0 & a_{33} & \cdots & a_{1m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & a_{nm} \end{bmatrix}}_{U}$$

A partir de uma matriz A, obtemos a solução do sistema com os seguintes passos [6]:

1. Obter a matriz U a partir da eliminação gaussiana
2. Obter a matriz L substituindo os valores dos multiplicadores em suas respectivas posições
3. Igualar o produto entre a matriz U e o vetor solução x com um vetor de incógnitas genérico y

$$A \cdot x = b \rightarrow \underbrace{L \cdot U}_{A} \cdot x = b \rightarrow L \cdot \underbrace{U \cdot x}_{y} = b$$

4. Resolver $L \cdot y = b$ para então obter y

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix}$$

5. Resolver $U \cdot x = y$ para obter o vetor solução x

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ 0 & a_{22} & a_{23} & \cdots & a_{2m} \\ 0 & 0 & a_{33} & \cdots & a_{1m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & a_{nm} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}$$

A vantagem adquirida ao fatorar a matriz A nestas duas matrizes consiste na obtenção facilitada de um vetor solução se apenas os valores do vetor dos termos independentes mudar, logo a solução para o novo sistema formado pela mesma matriz dos coeficientes porém um novo vetor b_m é obtida rapidamente [6].

A resolução de sistemas lineares de maneira computacional pode apresentar erros provenientes de divisão por números pequenos, erros de arredondamento, erros de truncamento, além existir o fator relacionado ao condicionamento das matrizes.

Uma maneira de combater os erros de arredondamento é através da utilização de **pivoteamento parcial**, que consiste em permutar linhas a partir do maior valor absoluto da variável líder antes de dar início a cada fase de eliminação [3]. O uso de pivôs de valor pequeno pode trazer instabilidade numérica e gerar resultados completamente errados. Para matrizes positivas definidas, o pivoteamento não é requerido mesmo que pivôs pequenos apareçam. No entanto, matrizes mal condicionadas apresentam problemas com o resultado obtido, sendo este inevitavelmente sensível [5].

Visando a implementação computacional deste método, utilizou-se o algoritmo 3 para obter as matrizes fatoradas L e U.

Algorithm 3: Decomposição LU com pivoteamento parcial

Data: Matriz A_{mxn}
for $i = 1, ..., n$ **do**
  $p_i = i$
end
for $k = 1, ..., (n-1)$ **do**
  $pivo = a_{kk}$
  $linhapivo = k$
  for $i = (k+1, ..., n$ **do**
    if $|a_{ik}| > |pivo|$ **then**
      $pivo = a_{ik}$
      $linhapivo = i$
    end
  end
end
if $linhapivo \neq k$ **then**
  $troca = p_k$
  $p_k = p_{linhapivo}$
  $p_{linhapivo} = troca$
  for $j = 1, ..., n$ **do**
    $troca = a_{kj}$
    $a_{kj} = a_{linhapivoj}$
    $a_{linhapivoj} = troca$
  end
end
for $i = (k+1), ..., n$ **do**
  $m = \dfrac{a_{ik}}{a_{kk}}$
  $a_{ik} = m$
  for $j = (k+1), ..., n$ **do**
    $a_{ij} = a_{ij} - m \cdot a_{kj}$
  end
end

Já visando obter o vetor solução, o seguinte algoritmo foi desenvolvido computacionalmente.

Algorithm 4: Resolução do sistema $A \cdot x = b$ pela decomposição LU

$c = p \cdot b \rightarrow$ Para computar as permutações no vetor dos termos independentes
for $i = 1, ..., n$ **do**
  $aux = p_i$
  $c_i = b_{aux}$
end
$L \cdot y = P \cdot b \rightarrow$ Para obter o vetor y, onde $U \cdot x = y$
for $i = 1, ..., n$ **do**
  $s = 0$
  for $j = 1, ..., (n-1)$ **do**
    $= s + a_{ij} \cdot y_j$
  end
  $y_i = c_i - s$
end
$U \cdot x = y \rightarrow$ Para obter o vetor solução x
for $i = n, ..., 1$ **do**
  $s = 0$
  for $j = (i+1), ..., n$ **do**
    $s = s + a_{ij} \cdot x_j$
  end
  $x_i = \dfrac{(y_i - s)}{a_{ii}}$
end

3.1.3 Decomposição de Gauss-Jordan

Este método possui este nome devido ao matemático alemão que difundiu a eliminação gaussiana e ao engenheiro alemão Wilhelm Jordan que popularizou este método através de seu livro sobre geodésia, a ciência que analisa a forma e dimensão da Terra [3].

A partir da matriz escalonada obtida através do método de eliminação gaussiana é possível realizar mais operações elementares para que o vetor solução seja obtido de forma imediata [3]. Para isto, é necessário:

1. Multiplicar cada linha não nula pelo inverso do pivô para obter 1 como variável líder
2. Zerar as posições acima do pivô através das operações elementares

Na eliminação gaussiana, a fase de eliminação é realizada da esquerda para direita objetivando obter zeros abaixo da posição de pivô. Já no método de Gauss-Jordan, a fase de eliminação partindo de uma matriz já escalonada ocorre da direita para esquerda com objetivo de obter zeros acima da posição de pivô. O processo de obter termos nulos acima e abaixo da posição de pivô caracteriza o método de Gauss-Jordan, onde as variáveis em posição de pivô, ou seja, as variáveis líderes apenas aparecem nestas posições, o que torna o processo de substituição mais simples, quando este é necessário (para o caso de matrizes não quadradas) [3].

Dada uma matriz A_{mxn} ampliada, após a aplicação do método desta seção obtemos uma matriz na forma escalonada reduzida:

$$A_{mxn}|b_m = \begin{bmatrix} 1 & a_{12} & 0 & \cdots & 0 & b_1 \\ 0 & 0 & 1 & \cdots & 0 & b_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & b_m \end{bmatrix}$$

Observa-se na matriz acima que o único valor não nulo nas colunas dos pivôs é o próprio pivô sendo o mesmo é igual a 1. Também observa-se o termo a_{12} presente em uma coluna onde não existe pivô, logo é permitido que este termo seja não nulo [3].

Dada uma matriz A_{mxn}, um fato importante a ser mencionado é que a mesma pode ser equivalente a diversas matrizes escalonadas, o que depende diretamente das operações elementares utilizadas para tal, no entanto, esta matriz A será equivalente a apenas uma matriz escalonada reduzida [6].

Algorithm 5: Método de Gauss-Jordan

```
for j = 1, ..., n − 1 do
    for i = j + 1, ..., m do
        m_ij = a_ij / m_jj   b_i = b_i − m_ij · b_j
        for k = j, ..., n do
            a_ik = a_ik − m_ij · a_jk
        end
    end
end
```

3.2 Métodos Iterativos

Na teoria, os métodos diretos de eliminação podem ser utilizados para resolver qualquer sistema linear de equações. No entanto, quando os sistemas a serem resolvidos são muito grandes, estes métodos diretos começam a apresentar problemas devido ao erro de arredondamento [3].

Em sistemas lineares muito grandes, mesmo quando o erro de arrendondamento é controlado, os métodos de eliminação podem não ser eficientes o suficiente para serem práticos. Neste caso, são utilizados os métodos iterativos cuja solução de um sistema é encontrada por meio de uma sequência de aproximações. Estes métodos não sofrem do problema de arredondamento descrito anteriormente nesta seção, no entanto, só podem ser utilizados em matrizes quadradas, ou seja, quando o número de variáveis é igual ao número de equações. Dizemos quando que o método diverge quando ele falha em obter uma solução, e converge quando a procura é bem sucedida [3].

3.2.1 Jacobi

Este método iterativo recebe este nome devido ao matemático alemão Karl Gustav Jacobi [3]. Em um sistema $A \cdot x = b$, divide-se a matriz dos coeficientes em três matrizes $A = L + D + U$, onde [7]:

- D é a diagonal principal de A;
- L é a matriz triangular inferior, apenas com os elementos infradiagonais de A;
- U é a matriz triangular superior, apenas com os elementos supradiagonais de A.

A partir desta divisão, temos:

$$x^{(k+1)} = D^{-1} \cdot (b - L + U) \cdot x^k \tag{2}$$

Expressando a Eq. (2) na forma desenvolvida para cada incógnita, temos:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{\substack{j=1 \\ j<i}}^{i-1} a_{ij}x_j^k - \sum_{\substack{j=i+1 \\ j>i}}^{n} a_{ij}x_j^k) \tag{3}$$

Para resolver um sistema a partir do método de Jacobi, partimos de uma solução inicial, isolamos a i-ésima incógnita x_i para cada i-ésima equação i e aplicamos a Eq. (3) [7].

$$x^{(0)} = \left\{x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}\right\}$$

Para implementação computacional do método, o algoritmo 6 foi utilizado.

Algorithm 6: Método iterativo de Jacobi

$iteracao = 0$
$iteracao = iteracao + 1$
$a = 0$; $b = 0$ Usados para avaliar o critério de parada
$y_i = 0 \rightarrow x^{(k+1)}$ Iteracao atual
for $i = 1, ..., n$ **do**
  if $i \neq j$ **then**
    $y_i = y_i + a_{ij} \cdot x_j$
  end
end
$y_i = \frac{1}{a_{ii}} \cdot (b_i - y_i)$
if $a < |y_i - x_i|$ **then**
  $a = |y_i - x_i|$
end
if $b < |y_i|$ **then**
  $b = |y_i|$
end
while $\frac{a}{b} < \varepsilon$ **do**
  $x = y$ $x = x^{(k)}$ é a solucao anterior e $y = x_{(k+1)}$ é a solucao atual
end

Descrito o método de Jacobi, o mesmo funciona a partir de uma sequência de aproximações até que a solução aproximada alcance um valor pré-determinado por um critério de parada. Os critérios mais comuns são [7]:

1. Máxima diferença absoluta entre os valores da iteração nova e da iteração anterior, entre todas as incógnitas
$$max \left\{ \left| x_i^{(k+1)} - x_i^{(k)} \right| \right\} \leq \varepsilon \Rightarrow i = 1, 2, ..., n$$

2. Máxima diferença relativa entre os valores da iteração nova e da iteração anterior, entre todas as incógnitas
$$max \left\{ \left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right| \right\} \leq \varepsilon \Rightarrow i = 1, 2, ..., n$$

3. Maior resíduo entre todas as equações, sendo este capaz de gerar valores inconsistentes para sistemas mal condicionados
$$max \left\{ \left| r_i^{(k+1)} \right| \right\} \leq \varepsilon \Rightarrow i = 1, 2, ..., n$$

$$r_i^{(k+1)} = b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k+1)}$$

3.2.2 Gauss-Seidel

Este método foi previamente descoberto por Carl Friedrich Gauss, que o descartou por não considerá-lo útil. No entanto, o matemático alemão Ludwig Philipp von Seidel descobriu de forma independente este método e o publicou após a morte de Gauss, cujo nome foi incluso na publicação [3].

Semelhante ao método de Jacobi, cujo valor obtido na resolução da Eq. (3) é utilizado em todo o conjunto de equações a cada iteração sendo as variáveis atualizadas ao mesmo tempo, no método de Gausss-Seidel as variáveis utilizam o valor mais atualizado possível [3].

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{\substack{j=1 \\ j<i}}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{\substack{j=i+1 \\ j>i}}^{n} a_{ij}x_j^k) \tag{4}$$

Como podemos observar na Eq. (4), os valores de x_j mais atualizados são utilizados, ou seja, são utilizados os valores calculados de $x_j^{(k+1)}$ das equações com $j < i$ dentro da iteração em andamento. Já na Eq. (3), os valores utilizados nesta mesma condição são referentes a $x_j^{(k)}$.

Visando a implementação computacional, o algoritmo 7 foi utilizado.

Algorithm 7: Método iterativo de Gauss-Seidel

1 Realizar os passos do algoritmo de Jacobi até a linha 16
2 $x = y$ onde $x = x^{(k)}$ é a solucao anterior e $y = x_{(k+1)}$ é a solucao atual
3 **until** $\frac{a}{b} < \varepsilon$

4 Matrizes especiais

O problema proposto para este estudo consiste na consideração de um sistema linear de ordem 15, $A \cdot x = b$, cuja solução exata corresponde ao um vetor com todos os componenentes iguais a 1. Para a matriz dos coeficientes A, foram consideradas as matrizes especiais descritas nesta seção.

4.1 Matriz de Cauchy

A matriz de Cauchy [8] é definida como uma matriz mxn atribuída ao somatório dos parâmetros $m+n$, $(x_1, ..., x_m)$ e $(y_1, ..., y_n)$ como pode ser visto a seguir:

$$C_i = \left[\frac{1}{x_i + y_j}\right] \Rightarrow i = 1, ..., m \therefore j = 1, ..., n \tag{5}$$

Onde x_i, y_j são números reais tais que $x_i + y_j \neq 0$ [9].

4.2 Matriz de Hilbert

A matriz de Hilbert [8, 9] é um caso da matriz de Cauchy, onde $x_i = i$ e $y_j = j - 1$ logo seus termos são definidos conforme Eq. (6):

$$H_n = \left[\frac{1}{i + j - 1}\right] \tag{6}$$

4.3 Matriz de Vandermonde

A matriz de Vandermonde é uma matriz quadrada composta por linhas que são potências de números fixos, com a seguinte forma [9]:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}$$

4.4 Matriz de Toeplitz

A matriz de Toeplitz possui seus termos a_{ij} constantes em suas diagonais e determinados por $T_{i,j} = t_{j-i}$ [10], possuindo a seguinte forma:

$$T = \begin{bmatrix} t_0 & t_1 & \cdots & t_{j-i} \\ t_{-1} & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{j-i} & \cdots & t_{-1} & t_0 \end{bmatrix}$$

5 Experimentos numéricos

Tendo conhecimento dos métodos expostos na seção 3, os mesmos foram desenvolvidos computacionalmente a partir dos algoritmos descritos na mesma seção. Logo após, as matrizes especiais descritas na seção 4 foram estudadas e comparadas objetivando entender qual o método mais vantajoso para resolução de cada matriz. Este estudo foi desenvolvido em uma máquina com as seguintes configurações: processador Intel i7 6600U, memória RAM de 16Gb, sistema operacional Windows 10 Pro x64 e SSD de 250Gb. Foi utilizada a linguagem Python para o desenvolvimento dos algoritmos.

Inicialmente, a resposta exata foi obtida a partir da expressão $A \cdot x = b$, onde A é uma matriz especial, x é o meu vetor com todos os componentes iguais a 1 e b é o vetor dos termos independentes a ser obtido.

Tendo estes resultados em mãos b_{exato}, partimos em busca de novos vetores das incógnitas x para cada método e para cada matriz especial. Então a equação $A \cdot x = b_{exato}$ foi proposta para cada matriz, sendo A as matrizes de Cauchy, Hilbert, Vandermonde e Toeplitz.

Para aferir os erros, utilizou-se o erro relativo médio do vetor solução, conforme Eq.(7) [11]:

$$ErroRelativo = \frac{|x - \bar{x}|}{|\bar{x}|}, x \neq 0 \tag{7}$$

Os resultados obtidos após a implementação de todos os métodos para todas as matrizes são expostos conforme as tabelas abaixo. Para os métodos iterativos, utilizou-se duas situações: vetor solução inicial com todos os elementos iguais a zero e número de iterações igual a 200; vetor solução inicial com todos os elementos iguais a 1 e número de iterações igual a 1. Optou-se por avaliar os métodos iterativos desta maneira pois mesmo com um grande número de iterações, o método não convergiu para nenhuma das matrizes, alcançando valores próximos ao número máximo disponível para armazenamento do tipo float64 na linguagem Python. Em alguns casos, quando o número de iterações superou 500, ocorreu overflow e não foi possível dar continuidade na avaliação do método iterativo.

Tabela 1: Matriz de Cauchy

Método	Tempo Computacional [s]	Erro Relativo Médio	Resíduo	Iterações
Eliminação Gaussiana	0,008976	1,5139	$9{,}69\text{x}10^{19}$	x
Decomposição LU	0,001994	1,2389	$1{,}03\text{x}10^{-15}$	x
Gauss-Jordan	0,003989	308,7169	1805,84	x
Jacobi	0,184842	$3{,}30\text{x}10^{295}$	infinito	200
Jacobi	0,008975	$1{,}35\text{x}10^{-15}$	7,6778	1
Gauss-Seidel	0,376918	$1{,}94\text{x}10^{296}$	infinito	200
Gauss-Seidel	0,002991	$1{,}45\text{x}10^{-15}$	7,6778	1

Para a matriz de Cauchy, o método que apresentou um melhor desempenho em relação ao tempo computacional foi a decomposição LU. Já sobre o erro relativo médio, os métodos iterativos cujo vetor solução inicial escolhido foi um vetor com todos os elementos iguais a 1 foi o que apresentou o menor erro, visto que é conhecido que este vetor é a solução exata do problema. No entanto, como o objetivo do método não é este e sim partir de um vetor qualquer podendo ou não convergir para a solução, este resultado não será

considerado o melhor. Para isto, a partir das implementações e resultados obtidos para este estudo, para a matriz de Cauchy o método que apresentou menor tempo computacional, erro e resíduo foi o método de decomposição LU.

Tabela 2: Matriz de Hilbert

Método	Tempo Computacional [s]	Erro Relativo Médio	Resíduo	Iterações
Eliminação Gaussiana	0,013983	2,5408	$4{,}58\text{x}10^{-16}$	x
Decomposição LU	0,001995	2,8937	$1{,}21\text{x}10^{-15}$	x
Gauss-Jordan	0,003989	743,3920	4819,50	x
Jacobi	0,176526	$4{,}39\text{x}10^{292}$	infinito	200
Jacobi	0,007978	$1{,}36\text{x}10^{-15}$	11,7471	1
Gauss-Seidel	0,377708	$5{,}29\text{x}10^{292}$	infinito	200
Gauss-Seidel	0,001994	$1{,}76\text{x}10^{-15}$	11,7471	1

Quando avaliados o desempenho dos métodos para a matriz de Hilbert, observa-se também um menor tempo computacional no método de Gauss-Seidel, que se assemelha ao tempo da decomposição LU. Para esta matriz, o método cujo tempo computacional é menor, assim como o erro, mesmo possuindo um resíduo grande (fato devido ao mal condicionamento da matriz) é o de Gauss-Seidel.

Tabela 3: Matriz de Vandermonde

Método	Tempo Computacional [s]	Erro Relativo Médio	Resíduo	Iterações
Eliminação Gaussiana	0,008975	$1{,}00\text{x}10^{-7}$	$2{,}38\text{x}10^{-7}$	x
Decomposição LU	0,001994	42,5141	8,8955	x
Gauss-Jordan	0,003988	$1\text{x}10^{-7}$	4,6145	x
Jacobi	0,167585	$5{,}11\text{x}10^{148}$	infinito	200
Jacobi	0,007979	$1{,}48\text{x}10^{-17}$	$1{,}7872\text{x}10^{17}$	1
Gauss-Seidel	0,374994	$1{,}46\text{x}10^{49}$	$1{,}97\text{x}10^{57}$	200
Gauss-Seidel	0,002990	$1{,}48\text{x}10^{-17}$	$1{,}78\text{x}10^{17}$	1

Para a matriz de Vandermonde, a matriz que possui progressões geométricas, o método que teve o melhor tempo computacional também foi o método da decomposição LU. No entanto, considerando o tempo computacional, o erro relativo médio e o resíduo, o método que apresentou melhor desempenho para a resolução do sistema linear cuja matriz dos coeficientes é esta matriz especial foi o método da eliminação gaussiana.

Tabela 4: Matriz de Toeplitz

Método	Tempo Computacional [s]	Erro Relativo Médio	Resíduo	Iterações
Eliminação Gaussiana	0,008975	$1{,}62\text{x}10^{-7}$	$3{,}55\text{x}10^{-15}$	x
Decomposição LU	0,002992	$1{,}55\text{x}10^{-16}$	$7{,}7429\text{x}10^{-15}$	x
Gauss-Jordan	0,003990	$2{,}66\text{x}10^{-16}$	$4{,}6145\text{x}10^{16}$	x
Jacobi	1,072521	1	$2{,}9859\text{x}10^{-156}$	200
Jacobi	0,007978	$1{,}70\text{x}10^{-16}$	133,5923	1
Gauss-Seidel	0,364651	1	0	200
Gauss-Seidel	0,002990	$1{,}70\text{x}10^{-16}$	133,5923	1

A matriz de Toeplitz apresentou bons resultados para todos os métodos, tanto em tempo computacional quanto em erro relativo médio, assim como na avaliação dos resíduos. Avaliando o tempo computacional, os métodos de decomposição LU e Gauss-Seidel como uma iteração apresentaram tempos menores e semelhantes. Em relação ao erro relativo, o menor erro foi obtido pelo método de decomposição LU. E em relação ao resíduo, o menor erro foi obtido para o método de Jacobi com uma iteração, no entanto, avaliar o resíduo

não apresenta um resultado confiável visto que as matrizes especiais observadas neste estudo são matrizes mal condicionadas, ou seja, sensíveis a qualquer variação nos parâmetros do sistema linear. Logo, para a matriz de Toeplitz o método que apresentou o melhor resultado foi o método da decomposição LU.

6 Conclusão

O objetivo deste estudo foi avaliar o comportamento dos métodos de resolução de sistemas lineares quando a matriz dos coeficientes A é uma matriz especial mal condicionada como as matrizes de Cauchy, Hilbert, Vandermonde e Toeplitz descritas neste trabalho.

A partir da implementação dos algoritmos na linguagem Python, quando avaliados o tempo computacional, erro relativo médio e resíduo, os seguintes métodos foram melhores para cada matriz:

- Matriz de Cauchy: Decomposição LU
- Matriz de Hilbert: Gauss-Seidel
- Matriz de Vandermonde: Eliminação Gaussiana
- Matriz de Toeplitz: Decomposição LU

É importante mencionar que estes resultados variam conforme muitos aspectos, como: implementação dos algoritmos, máquina utilizada para obtenção dos resultados, experiência do aluno com a linguagem adotada, assim como a construção das matrizes.

Visando melhorar os resultados e obter até mesmo respostas diferentes para o melhor método para cada matriz especial, sugere-se manipular e tratar a matriz antes da utilização dos métodos de resolução, seja para torná-la diagonal dominante ou avaliar o seu número de condicionamento. Também sugere-se o refino da implementação dos algoritmos em uma linguagem de programação e a utilização de computação paralela.

Referências

[1] Fabio Barros de Sousa; Elizabeth Rego Sabino; Elizete Rego Sabino. Abordagem histórica e conceitual sobre os sistemas de equações lineares e sua relação com matrizes e determinantes. *III Jornada de Estudos em Matemática*, page 15, Outubro de 2017.

[2] J. L. Boldrini. *Álgebra Linear*. Harper and Row, 1980.

[3] Jeffrey Holt. *Linear Algebra with applications*. W. H. Freeman, 2012.

[4] Seymour Lipschutz; Marc Lipson. *Linear Algebra*. Mc Graw Hill, 2009.

[5] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley - Cambridge Press, 2016.

[6] Diana Sasaki. Notas de aula da disciplina Álgebra Linear. 2021.

[7] Sérgio Peters; Julio Felipe Szeremeta. *Cálculo Numérico Computacional*. Editora UFSC, 2018.

[8] Miroslav Fiedler. Notes on hilbert and cauchy matrices. *Linear Algebra and its Applications*, pages 351–356, 2009.

[9] Marcos Savitraz. Determinante de algumas matrizes especiais, 2015.

[10] Maria Gabriela Eberle e Maria Cristina Maciel. Finding the closest toeplitz matrix. *Computational and Applied Mathematics*, 22(1):1–18, 2003.

[11] Dagoberto Adriano Rizzotto Justo; Esequia Sauter; Fabio Souto de Azevedo; Leonardo Fernandes Guidi; Pedro Henrique de Almeida Konzen. Cálculo Numérico. Um livro colaborativo - Versão Python, Agosto de 2020.